4. 分割(partitioning)と
分割統治法(divide-and-conquer)

4.1 分割法(1)

問題を幾つかの部分に単純に分割してそれぞれの部分を別々に計算する。

分割戦略

(1)データ分割法(領域分割)

- データを分割して分割されたデータに対して並行に操作を行う
- 並列処理の主戦略で，適用事例が多い

(2)機能分割

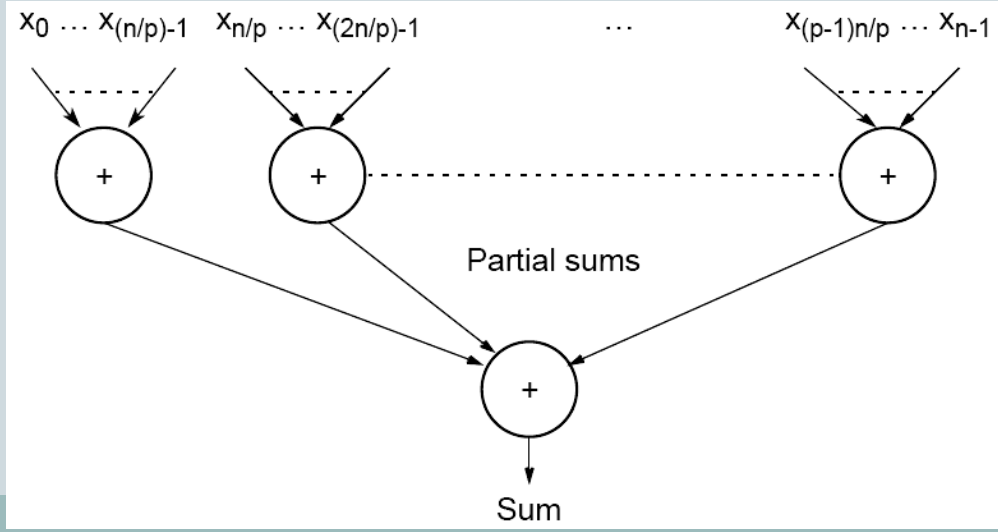
- プログラムを独立した機能に分割してそれぞれの機能を並列実行する
- それほど事例は多くない

4.1 分割法(2)

データ分割の例

数列 $x_0, x_1, \ldots, x_{n-1}$の加算を考えよ。

1. n/m個の数からなるm個の部分に分ける $(x0, \ldots, x_{n/m-1}), (x_{n/m}, \ldots, x_{2n/m-1}), \ldots, (x_{(m-1)n/m} \ldots \ x_{n-1})$
2. m台のプロセッサで独立に加算を行ってm個の部分和を作る
3. m個の部分和を一台のプロセッサで加算する



4.1 分割法(3)

Master:

```
s=n/m;
for(i=0, x=0; i<m; i++, x=x+s)
    send(&numbers[x], s, Pi);
sum=0;
for(i=0; i<m; i++){
    recv(&part_sum, P_any);
    sum= sum+part_sum;
}
```

Slave:

```
recv(numbers, s, Pmaster);
part_sum=0;
for(i=0; i<s; i++)
    part_sum = part_sum + numbers[i];
send(&part_sum, Pmaster);
```

4.1 分割法(4)

Master:

```
s=n/m;
bcast(numbers, s, Pslave_group);
sum=0;
for(i=0; i<m; i++){
  recv(&part_sum, Pany);
  sum = sum + part_sum;
}
```

Slave:

```
bcast(numbers, s, Pmaster);
start = slave_number*s;
end=start+s;
part_sum=0;
for(i=start; i<end; i++)
  part_sum=part_sum+numbers[i];
send(&part_sum, Pmaster);
```

slave_numberは、プロセスIDに対応する。

4.1 分割法(5)

scatter(スキャター): 分配送信

reduce_add: 簡約ルーチン

Master:

```
1.   s=n/m;
2.   scatter(numbers, &s, Pgroup, root=master);
3.   reduce_add(&sum, &s, Pgroup, root=master);
```

Slave:

```
1.   scatter(numbers, &s, Pgroup, root=master);
2.   ...
3.   reduce_add(&part_sum, &s, Pgroup, root=master);
```

4.1 分割法(6)

解析

逐次計算の時間計算量:

- n-1回の加算 -> O(n)

並列計算:

- 段階1-通信
  - m個のプロセスがそのn/m個の数を読み出す。

    $t_{comm1} = m(t_{startup} + (n/m)t_{data})$
  - スキャタを用いた場合、起動時間の回数を減らせる可能性がある。

    $t_{comm1} = t_{startup} + n\, t_{data}$
    - 実装によることに注意！
- 段階2-計算
  - スレーブプロセスのn/m個の数の加算

    n/m – 1 回
  - m個のプロセスは同時に動作するので、並列計算時間は

    $t_{comp1} = n/m - 1$

4.1 分割法(7)

- 段階3−通信
  - 個別の送受信ルーチンを用いて結果を返すための通信時間は

    t_comm2 = m(t_startup + t_data)
  - ギャザー(収集)と簡約を用いれば(但し，実装による)

    t_comm2 = t_startup + m t_data
- 段階4−計算
  - 最後の累積のためにマスタはm個の部分和の加算を行う。

    t_comp2 = m−1
- 全体

  t_p = (t_comm1+t_comm2)+(t_comp1+t_comp2)

  = (m(t_startup+(n/m)t_data))+(m(t_startup+t_data))+ (n/m−1+m−1)

  = 2m t_statup + (n+m)t_data + m+n/m−2

  = O(n+m)
- 逐次処理O(n)と並列処理O(n+m)
- 通信量を無視すれば　$S = t_s/t_p$ = (n−1)/(m＋n/m−2)
- nが大きいとき速度向上度はmに近づく

4.1 分割法(8)

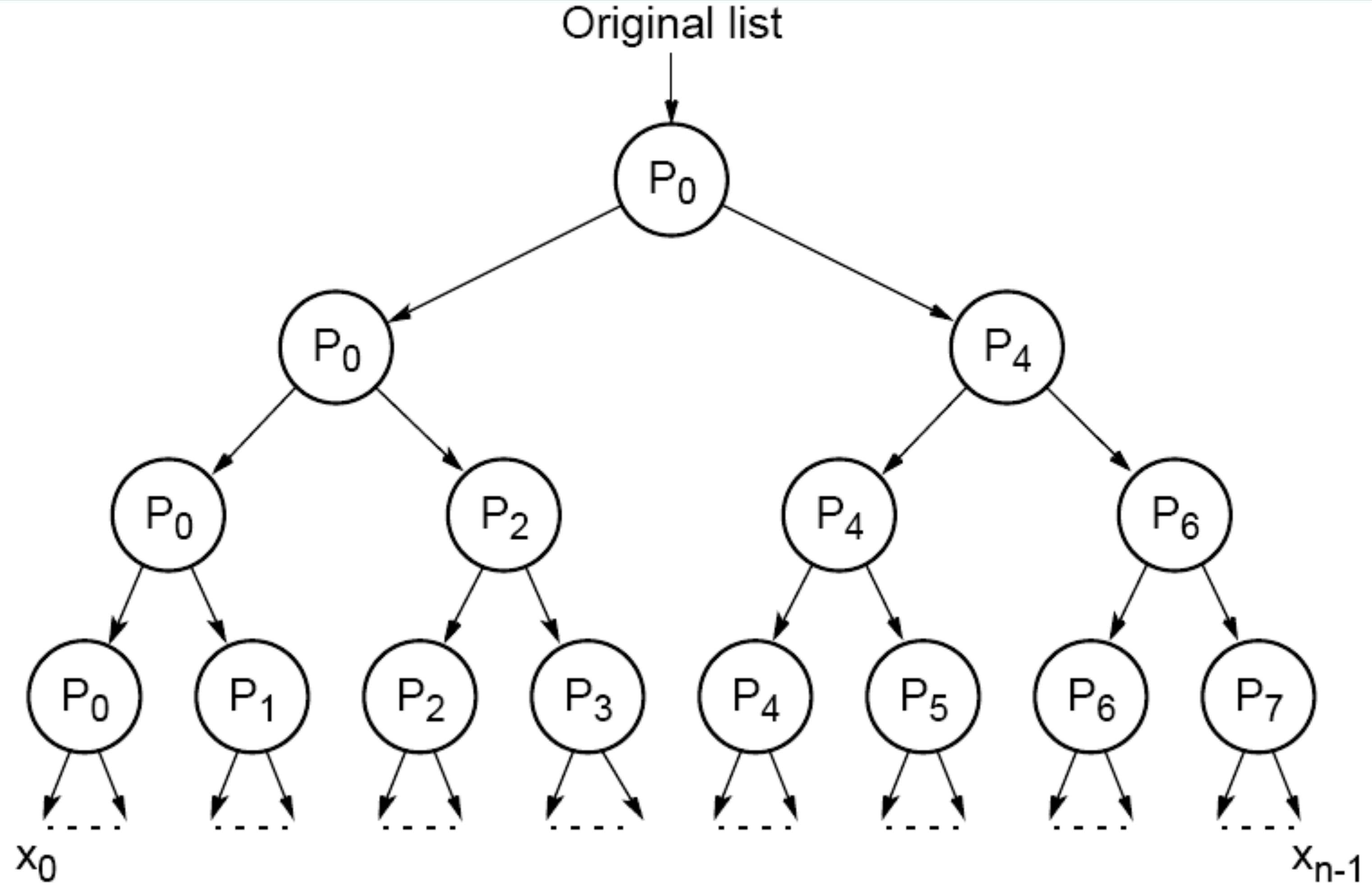
分割統治法

- 問題をもとのものと同じ形のより小さい問題に分割する
- 分割した一方を自分自身で処理し，残りを別のプロセスに割当てる

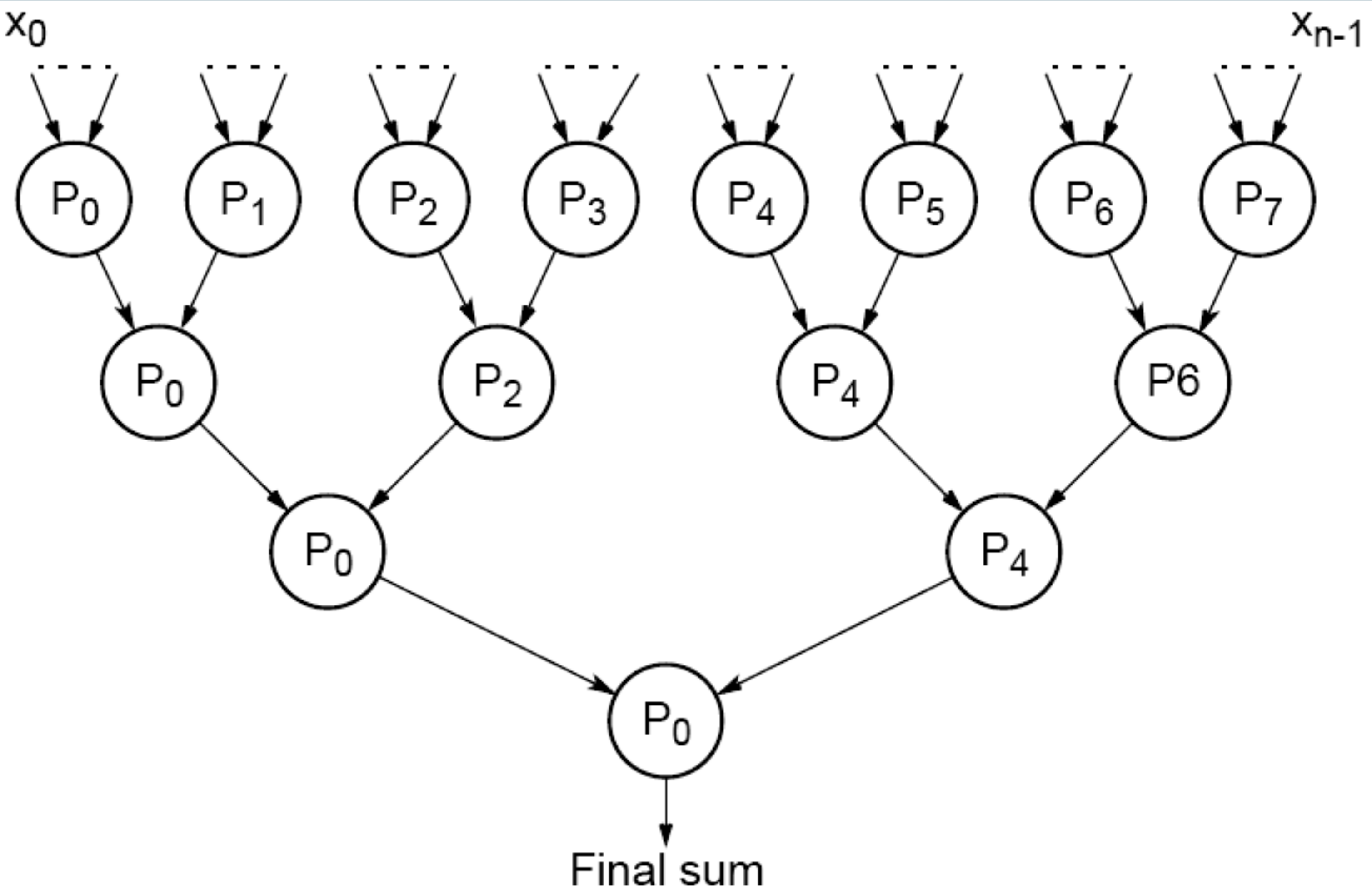
数列の加算の例：

- 各プロセッサはリストの半分を保持し，残りを別のプロセスに渡す
- pプロセッサの場合，各リストのサイズはn/pとなり，2分木の高さはlog pとなる

4.1 分割法(9-1)



4.1 分割法(9-2)



4.1 分割法(9-3)

P0:

```
divide(s1,s1,s2);
send(s2,P4);
divide(s1,s1,s2);
send(s2,P2);
divide(s1,s1,s2);
send(s2,P1);
part_sum = *s1;
recv(&part_sum1, P1);
part_sum = part_sum + part_sum1;
recv(&part_sum1, P2);
part_sum = part_sum + part_sum1;
recv(&part_sum1, P4);
part_sum = part_sum + part_sum1;
```

4.1 分割法(9-4)

P4:

```
recv(s1, P0);
divide(s1,s1,s2);
send(s2,P6);
divide(s1,s1,s2);
send(s2,P5);
part_sum = *s1;
recv(&part_sum1, P5);
part_sum = part_sum + part_sum1;
recv(&part_sum1, P6);
part_sum = part_sum + part_sum1;
send(&part_sum, P0);
```

4.1 分割法(10)

解析

- 仮定
  - nは2の巾乗と仮定する．通信起動時間は無視する．
- 通信
  - 分割段階のステージ数はpプロセッサではlog pで、通信時間は

    t_comm1=(n/2)t_data+(n/4)t_data+…+(n/p)t_data=(n(n−1)/p)t_data
  - 組合わせ段階

    t_comm2 = t_data log p
- 全通信時間は

  t_comm = t_comm1 + t_comm2 =(n(p−1)/p)t_data+t_data log p + n/p + log p
  - pが一定ならO(n)となる
- 計算
  - 分割段階の最後にn/p個の数の加算が行われ，組合わせ段階では各ステージで1回の加算が行われる

    t_comp=n/p + log p
- 全体

  t_p = (n(p−1)/p)t_data + t_data log p + n/p + log p

4.2 分割統治法の例

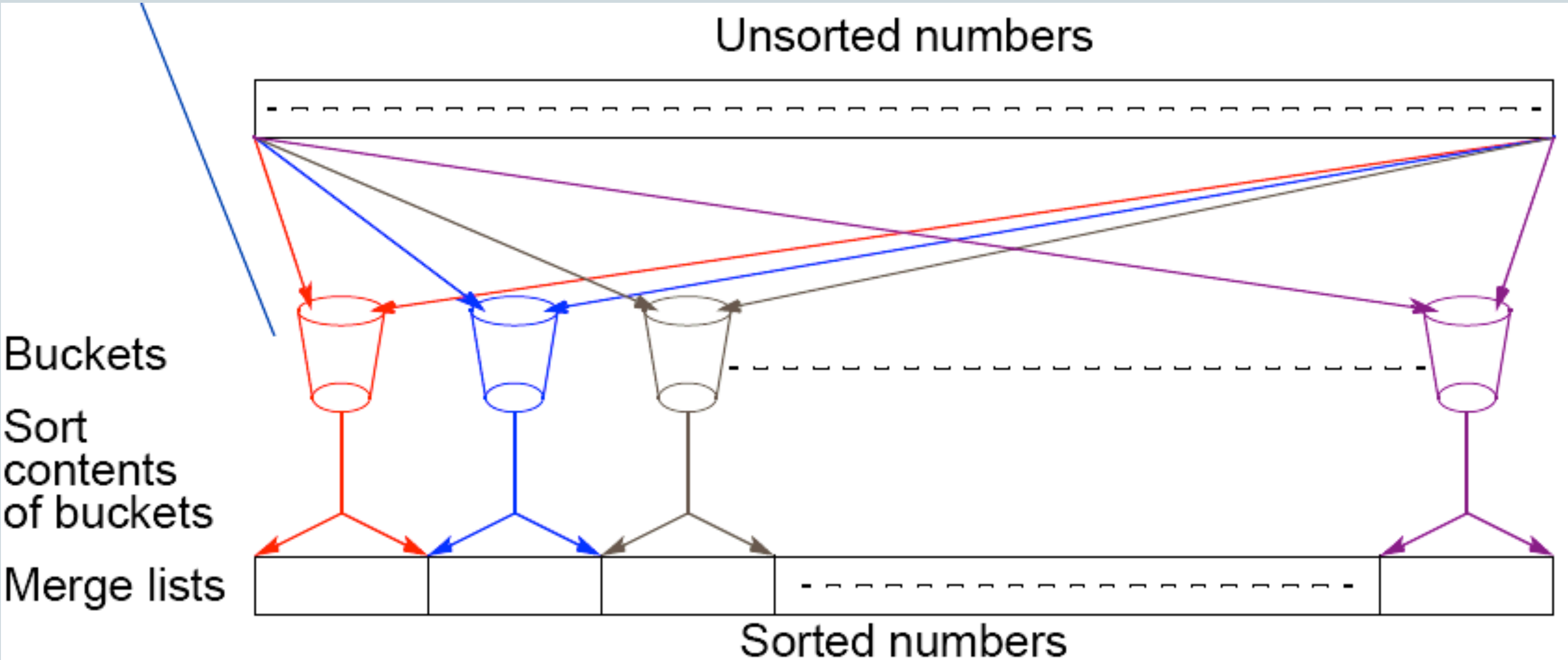
バケツソート法によるソート

- バケツソート
  - 比較交換ではなくて分割に基づくソート
  - 数が既知の区間，たとえば，0〜a-1の範囲に分布している時に可能
  1. 区間をm個の等しい領域，すなわち，0からa/m-1，a/mから2a/m-1,…に分割し，各領域にバケツを一つ割当てる。
  2. 入力された数を一つづつ該当するバケツに入れていく
  3. 各バケツの中をソートする
  4. 連結する
- 各領域のサイズが1であればステップ(3)は必要ない
- 逐次処理の時間計算量
- ステップ(2)は，O(n)
- ステップ(3)を比較に基づくアルゴリズムで処理すると

  O(n/m log n/m)
- ただし，数は一様に分布し各バケツに入るデータの数は均等としている
- 全体では

  t_s = n + m(n/m)log(n/m) = n + nlog(n/m) = O(n log(n/m))

4.2 分割統治法の例(2)



Unsorted numbers
Buckets
Sort
contents
of buckets
Merge lists
Sorted numbers

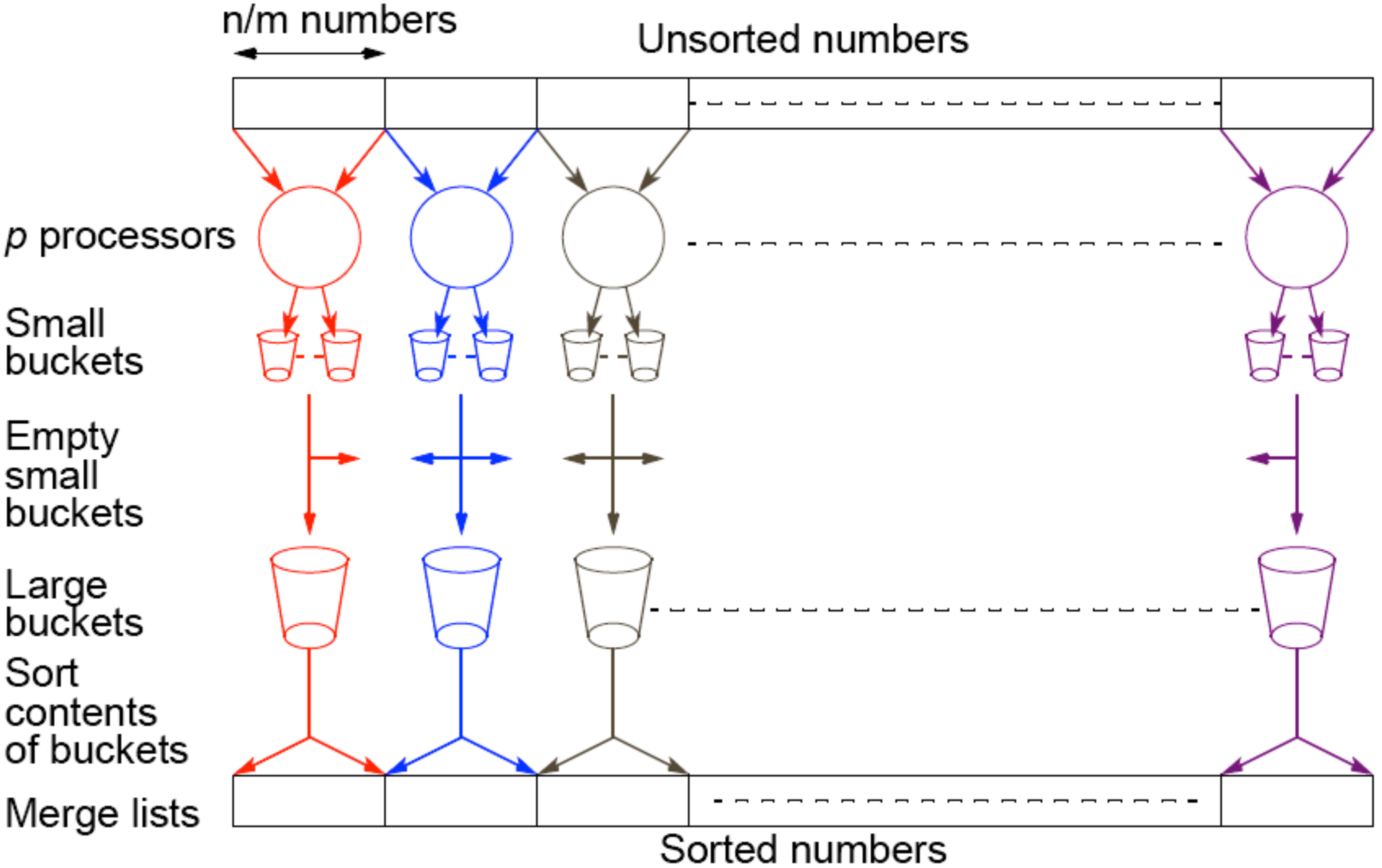
4.2 分割統治法の例(3)

並列アルゴリズム

- p台のプロセッサを準備する
- データをp個の領域に分割し，p台のプロセッサに割当てる
- 各プロセッサP_iはバケツB_iを保持すると同時にb_j (all $j \neq i$)を準備する

1. 割当てられたn/m個のデータをB_i or b_j に分類する
2. 各バケツb_jをプロセッサP_jに送信する
3. 大きいバケツをソート
4. 連結する

4.2 分割統治法の例(4)



n/m numbers
Unsorted numbers
p processors
Small buckets
Empty small buckets
Large buckets
Sort contents of buckets
Merge lists
Sorted numbers

4.2 分割統治法の例(2)

解析

- 段階1:計算および通信
  - n個の数をp個の領域に分割するのにnステップかかるとすると

    t_comp1 = n
  - n/p個のデータを含むp個の分割を各プロセッサに送るために，ブロードキャスト，あるいは，スキャタを用いると

    t_comm1 = t_startup + t_data * n

- 段階2:計算
  - n/p個の数をp個のバケツに分けるのに要する時間は

    t_comp2 = n/p

4.2 分割統治法の例(3)

解析

- 段階3:通信
  - 小さいバケツを送信する．各小バケツにおよそn/p^2個の数が入っていて，p-1個の小バケツを送信する

    $$t\_comm3 = p(p-1)(t\_startup + (n/p^2)t\_data)$$

  - 通信がオーバーラップされると仮定すると

    $$t\_comm3 = (p-1)(t\_startup + (n/p^2)t\_data)$$

- 段階4:計算
  - 大バケツが同時にソートされる

    $$t\_comp4 = (n/p)\log(n/p)$$

- 全体

  $$t\_p = t\_startup + t\_data\ n + n/p$$
  $$+ (p-1)(t\_startup +(n/p^2)t\_data) + (n/p)\log(n/p)$$